MORE GRANULAR AND MORE EFFICIENT WRITE PROTECTION FOR DISK VOLUMES

BACKGROUND OF THE INVENTION

The present invention relates generally to techniques for long term data archiving in a storage system. More particularly the present invention relates to a storage system and method for protecting data on a disk volume by managing a write pointer or a write protected area.

Conventionally, long term data archiving has been accomplished write once read many (WORM) storage media. Recently the need for long term data archiving has increased. This need has been made more acute, for example, by the passage of various regulations. These regulations include, for example, Regulations like SEC (Securities and Exchange Act) and 21 CFR (Code of Federal Regulations) Part 11 of the Food and Drug Administration act. These regulations require regulated companies to keep data for a long term. An important factor in such regulations is that the data must not be changed during the retention period. As the result, the data need to be stored on WORM storage media.

Logical device (LDEV) Guard disk subsystems have WORM capability. With this capability, if a volume is set to be write-protected, no one can write or change any data stored on the volume. Since data need not to be kept after an expiration of a period required by the regulations, LDEV guard provides a retention period for a volume. After expiration of the retention period, users can then write and change data on the volume. The storage system has an internal timer for this purpose.

However, some regulations require a strict WORM implementation where the WORM setting cannot be altered by anyone in the world. Similarly, such strict WORM implementation requires that the retention period or the internal timer in the storage system cannot be altered.

Further, LDEV does not allow for granularity of Write-Protected Area. Particularly, because LDEV guard protects data at a volume level, sometimes it's not useful. The typical size of a volume is several GBytes. On the other hand, the typical size of a file which users want to archive is tens of hundreds of MBytes. As the result, the volume can't be set to a write-protected state until the volume is filled by many files. During this period, there is no write-protection for files in the volume.

Still further, the large size of the data structure can be a problem. Because there are a large number of disk blocks in the disk array, when the storage system provides a write protection at a disk block level, the size of a data structure that tracks the write protected disk blocks and the retention period for these write protected disk blocks becomes very large. Sometime the size of the data structure requires a large amount of memory in the storage system, thereby increasing the cost of the storage system.

SUMMARY OF THE INVENTION

The present invention provides an apparatus, method and system for protecting data on a disk volume by managing a write pointer or a write protected area.

More particularly the present invention provides an apparatus, method and system for managing a next write pointer for each WORM-enabled

volume. According to the present invention the next write pointer indicates which disk block number on the volume a host can write data. If the disk block pointed to by the next write pointer was already written, then the next write pointer is incremented so as to point to the next disk block at which data has not been written by the host. Further according to the present invention a write request from a host is ignored or an error message is returned by the storage system if the disk block number indicated by the write request is less than the disk block number pointed to by the next write pointer.

The present invention provides another apparatus, method and system for managing a write-protected area. According to the present invention a write-protected area is identified by using a volume ID in a storage system, a beginning offset of the protected area in the volume, and an ending offset of the protected area in the volume. As per the present invention a write request from a host is ignored or an error message returned by a storage system, if an offset indicated by the write request is within one of write-protected areas. Further as per the present invention two or more write-protected areas are combined into one write-protected area, if these areas form a continuous area on the volume. When such combining is performed, the longest retention period within the combined write-protected areas is selected as the retention period for the combined write-protected areas.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and a better understanding of the present invention will become apparent from the following detailed description of example embodiments and the claims when read in connection with the accompanying

drawings, all forming a part of the disclosure of this invention. While the foregoing and following written and illustrated disclosure focuses on disclosing example embodiments of the invention, it should be clearly understood that the same is by way of illustration and example only and the invention is not limited thereto, wherein in the following brief description of the drawings:

Fig. 1 is a diagram for explaining a storage system with a WORM configuration table according to the present invention;

Fig. 2 is a first embodiment of a WORM configuration table according to the present invention;

Fig. 3 is a second embodiment of a WORM configuration table according to the present invention;

Fig. 4 is a flowchart illustrating a first embodiment of a procedure for processing a write request according to the present invention;

Fig. 5 is a flowchart illustrating a second embodiment of a procedure for processing a write request according to the present invention;

Fig. 6 is a flowchart illustrating a first embodiment of a procedure for decreasing a retention period according to the present invention;

Fig. 7 is a flowchart illustrating a second embodiment of a procedure for decreasing a retention period according to the present invention;

Fig. 8 is a diagram for explaining use of a next write pointer according to the present invention; and

Fig. 9 is a diagram for explaining use of a write protected area according to the present invention.

DETAILED DESCRIPTION OF THE INVENTION

The present invention as will be described in greater detail below provides an apparatus, method and system for managing a next write pointer for each WORM-enabled volume. According to the present invention as illustrated in Fig. 8, a next write pointer 801 indicates which disk block number on the volume 802 a host can write data. If the disk block pointed to by the next write pointer 801 was already written (Read Only Area 803), then the next write pointer 801 is incremented so as to point to the next disk block at which data has not been written by the host (Read and Write Area 804). Further according to the present invention a write request from a host is ignored or an error message is returned by a storage system if the disk block number indicated by the write request is less than the disk block number pointed to by the next write pointer 801.

The present invention provides another apparatus, method and system for managing a write-protected area. According to the present invention, as illustrated in Fig. 9, a write-protected area 901a, 901b is identified by using a volume ID in a storage system in a beginning offset 902a, 902b of the write protected area 901a, 901b and an ending offset 904a, 904b of the write protected area 901a, 901b in the volume 903. As per the present invention a write request from a host is ignored or an error message returned by a storage system, if an offset indicated by the write request is within one of write-protected areas 901a, 901b not the read and write areas 905a, 905b, 905c.

Further as per the present invention two or more write-protected areas are combined into one write-protected area, if these areas form a continuous

area on the volume. When such combining is performed, the longest retention period within the combined write-protected areas is selected as the retention period for the combined write-protected areas.

The present invention provides various embodiments as described below. However it should be noted that the present invention is not limited to the embodiments described herein, but could extend to other embodiments as would be known or as would become known to those skilled in the art.

Embodiment 1

The present invention provides a system configuration for the first embodiment as illustrated in Fig. 1, including a storage system 104, hosts 101a and 101b, and a management console 102.

The storage system includes multiple volumes 109a, 109b, 109c which are represented on storage media upon which data is stored and retrieved, an interface 105 which interfaces the storage system 104 with the hosts 101a, 101b, a disk controller 106 which controls the storage system 104, a cache memory 107 which temporarily stores data, an internal timer 110 which measures a period of time and a WORM configuration table 108 which stores information as to whether the volumes are WORM enabled. The details of the WORM configuration table 108 are discussed below.

The disk controller receives I/O requests from hosts 101a, 101b via a interface 105 and processes the requests. The disk controller accesses the cache memory 107 and the volumes 109a, 109b, 109c. The cache memory 107 caches some portions of data in the volumes 109a, 109b, 109c for faster access. The internal timer 110 measures a relative time and as such its value cannot be changed.

The hosts 101a, 101b run applications and conduct read/write of data from/to volumes 109a, 109b, 109c in a storage system via a link 103a, 103b as required by the applications. Typically Fibre channel, FICON, SCSI, Ethernet are used for the link 103a, 103b between a host 101a, 101b and a storage system 104. However, other types of links can be used.

The management console 102 is attached to a storage system 104 via a link 103c and provides management capabilities including configuring the WORM configuration table 108a in the storage system 104.

The first embodiment of the WORM configuration table 108a as illustrated in Fig. 2, includes various entries for each volume in a storage system 104. Particularly, the entries of the WORM configuration table indicate by use of a next write pointer which of a plurality of areas of the volume is write protected. The entries includes:

1) An identification of a volume 201,

2) An on/off indicator which indicates whether the volume is WORM-enabled 202.

3) An indication of block size at which data is write-protected 203.

4) An indication of the maximum number of blocks 204 (i.e. the capacity of the volume equals the block size times by the maximum number of blocks).

5) A next write pointer 205. The next write pointer has a block number and shows a host can read and write data between the block number indicated by the next write pointer and the maximum block number in a volume. On the other hand, the host can't write any data between the beginning of the block number, i.e. 0, and the block number indicated by the

next write pointer minus one. Fig. 8 as described above illustrates an example of using the next write pointer. The volume named "Vol0" has 0 to 1024 disk blocks and the next write pointer indicates "3". So an area between the block 0 to block 2 is a read-only area and the other area is a read and write area.

6) A retention period 206. This period shows how long data in a volume needs to be write-protected after the volume has been filled by data, i.e. the next write pointer points to the maximum block number plus one. The value of the retention period is decreased by one a day. The storage system uses its internal timer to determine if a day is passed. If more granular retention is required, hours or minutes, seconds can be used as a unit.

The management console 102 and a host 101a,b can configure the WORM configuration table 108a. The disk controller allows them the following configurations:

1) Create a new entry for a specified volume only if there is no entry for the volume in WORM configuration table.

2) Make an entry WORM-enabled only if the entry isn't WORM-enabled.

3) Delete an entry only if the retention period of the entry equals to zero.

4) Read entries in WORM configuration table.

Fig. 4 is a flowchart illustrating a first embodiment of a procedure for processing a write request using the first embodiment of the WORM configuration table 108a according to the present invention. Particularly the

flowchart of Fig. 4 illustrates processes to be conducted by the disk controller 106 if the disk controller 106 receives a write request from a host 101a,b.

Upon receipt of a write request the disk controller 106 checks if the write request is for any volume that is WORM enabled by using the WORM configuration table 108a. (Step 401). If write request is for a WORM enabled volume, then the process proceeds to Step 402. If write request is not for a WORM enabled volume, then the process proceeds to Step 405. The disk controller 106 checks if the offset specified by the write request is larger than or equal to the offset indicated by the next write pointer of the volume. (Step 402). If the offset specified by the write request is larger than or equal to the offset indicated by the next write pointer of the volume, then the process proceeds to Step 403. If the offset specified by the write request is not larger than or equal to the offset indicated by the next write pointer of the volume, then the process proceeds to Step 406. The disk controller 106 writes data in the request to the disk block pointed to by the next write pointer. (Step 403). The disk controller 106 increments a value of the next write pointer by one and exits this procedure. (Step 404) The disk controller 106 processes the write request and exits this procedure. (Step 405). The disk controller 106 returns an error message to the requesting host and exits this procedure. (Step 406).

Fig. 6 is a flowchart illustrating a first embodiment of a procedure for decreasing a retention period using the first embodiment of the WORM configuration table 108a according to the present invention.

According to the present invention the internal timer 110 of the storage system 104 sends an interrupt to the disk controller 106 once a day. The

timing can be changed like once an hour, once a minute, etc. Alternatively, the disk controller 106 checks the internal timer 110 to determine how much time has passed since it previously checked the internal timer 110. Whether the disk controller 106 receives an interrupt or a day has passed, the disk controller 106 executes the process/procedure illustrated in Fig. 6.

Specifically according to the present invention, for each entry in the WORM configuration table 108a, the disk controller 106 checks if the value of the next write pointer is larger than the max block number (i.e. the volume has been filled by files). (Step 601). If the value of the next write pointer is larger than the max block number, then the process proceeds to Step 602. If there is no such entry in WORM configuration table, the process exits the procedure. If the value of the next write pointer is larger than the max block number, then the disk controller 106 decreases the value of the retention period by one day. (Step 602). The disk controller 106 then checks if the value of the retention period is zero. (Step 603). If the value of the retention period is zero, then the process proceeds to Step 604. If the value of the retention period is not zero not, then the process proceeds back to Step 601. Thereafter, the disk controller 106 turns WORM-enable off for the entry (i.e. the retention period for data in the volume has been expired). (Step 604). Then the process proceeds back to Step 601.

Embodiment 2

The present invention provides a system configuration for the second embodiment the same as that provided for the first embodiment as illustrated in Fig. 1, including a storage system 104, hosts 101a and 101b, and a management console 102. One of the differences between the first

embodiment and the second embodiment is the WORM configuration table. The second embodiment provides a WORM configuration table 108b as illustrated in Fig. 3 which includes various entries for each volume in a storage system 104. These entries include:

1) An area ID (301) showing an identification of a write protected area.

2) A volume ID (302) in which the protected area is included.

3) A beginning offset (303) showing a beginning offset of the write protected area in the volume.

4) An ending offset (304) showing an ending offset of the write protected area in the volume.

5) A retention period (305) showing how long data in the protected area must remain read-only.

Fig. 9 illustrates an example of write-protected areas within a volume "Vol0". There are two areas, "Area0" and "Area1". Any areas specified by the write-protection areas cannot be written. The other areas are read and write areas.

A management console and a host can configure the WORM configuration table 108b. The disk controller 106 allows for the following configurations.

1) Create a new write protected area only if there is no offset-overlap among the new entry and entries in WORM configuration table.

2) Extend a retention period only if the new retention period is larger than the current retention period.

3) Delete an entry only if the retention period of the entry equals to zero.

4) Read entries in WORM configuration table.

5) Combine two or more protected areas into one protected area only if these areas are continuing. The largest retention period in the areas is selected as the combined write protected area.

Fig. 5 is a flowchart illustrating a second embodiment of a procedure for processing a write request using the second embodiment of the WORM configuration table 108b according to the present invention. Particularly the flowchart of Fig. 5 illustrates processes to be conducted by the disk controller 106 if the disk controller 106 receives a write request from a host 101a,b. Upon receipt of a write request the disk controller 106 sets temporary values including Top = an offset specified in the write request. Bottom = Top + the size of data in the write request. (Step 501). For each write-protected area in the WORM configuration table 108b having a retention period that is larger than zero, the disk controller 106 checks if the write request is directed to a volume specified in the write protected area. (Step 502). If the write request is directed to a volume specified in the write protected area, the process proceeds to Step 503. If the write request is not directed to a volume specified in the write protected area, then the process proceeds to Step 504. The disk controller 106 checks whether one of the following conditions is satisfied for the write-protected area (i.e. is there any overlap between the area to be written and the write protected area): (1) Beginning Offset <= Top <= Ending Offset, (2) Beginning Offset <= Bottom <= Ending Offset. (Step 503). If there is an overlap between the area to be written and the write protected area, then the process proceeds to Step 505. If there is not any overlap between the area to be written and the write protected area, then the

process proceeds to Step 502 to check another area. The disk controller 106 then processes the write request. (Step 504). Thereafter the disk controller 106 returns an error message to the requesting host. (Step 505).

Fig. 7 is a flowchart illustrating a second embodiment of a procedure for decreasing a retention period using the second embodiment of the WORM configuration table 108b according to the present invention.

According to the present invention the internal timer 110 of the storage system 104 sends an interrupt to the disk controller 106 once a day. The timing can be changed once an hour, once a minute, etc. Alternatively, the disk controller 106 checks the internal timer 110 to determine how much time has passed since it previously checked the internal timer 110. Whether the disk controller 106 receives an interrupt or a day has passed, the disk controller 106 executes the process/procedure illustrated in Fig. 7.

Specifically according to the present invention, for each entry in the WORM configuration table 108b, the disk controller 106 checks if the value of the retention period of the entry is larger than zero. (Step 701). If the value of the retention period of the entry is larger than zero, then the process proceeds to Step 702. If the value of the retention period of the entry is not larger than zero, then the process proceeds to exit the procedure. The disk controller 106 then decreases the value of the retention period by one day. (Step 702).

While the invention has been described in terms of its preferred embodiments, it should be understood that numerous modifications may be made thereto without departing from the spirit and scope of the present

invention. It is intended that all such modifications fall within the scope of the appended claims.